Robust Deep Reinforcement Learning Control for a Launcher Upper Stage Module with Stability Certificate

Périclès Cocaul, Sylvain Bertrand, Hélène Piet-Lahanier, Martine Ganet, Lori Lemazurier

Robust Deep Reinforcement Learning Control for a Launcher Upper Stage Module with Stability Certificate

Périclès Cocaul[1], Sylvain Bertrand[1], Hélène Piet-Lahanier[1], Martine Ganet[2,3] and Lori Lemazurier[2]

***Abstract*— This paper considers the design of an attitude controller of a launcher upper stage module during its exo-atmospheric phase, where short boosts are performed to adapt the flight path. Those maneuvers can increase propellants' motion in the tanks, leading to the so-called sloshing phenomenon that may affect the stability of the vehicle.**

A Deep Reinforcement Learning (DRL) algorithm is proposed to design the controller accounting for non linearities of the launcher and sloshing dynamics as well as presence of time-delay, bias and saturations on the actuation system. Based on Proximal Policy Optimization (PPO) and Almost Lyapunov functions in an actor-critic scheme, it allows to robustly learn a controller along with stability certificates, in presence of model uncertainties. Simulation results are proposed to illustrate the approach.

I. Introduction

Control design using Machine Learning (ML) is a soaring domain for the automatic community and especially for complex systems with partially unknown dynamics. Some practical implementations emerged such as motion planning with unknown dynamics [1] or the transient control of liquid rocket engines [2]. Systems with strenuous representation of the dynamics and subject to disturbing phenomena, such as launchers, are particularly concerned by these new possibilities which could drastically enlarge the performance domain. One of the main phenomenon to take into account for space vehicles is the motion of propellant within tanks, called sloshing. Its impact has been extensively studied [3], [4]. However, efficient methods for rejection of this perturbation such as structured H_∞ synthesis [5] still require good knowledge of the dynamics which is seldom available. In addition, complexity of the involved dynamics make the synthesis of an analytical control law a difficult task. Nevertheless, if available, the complex dynamic model can be used in simulation to generate numerical data that can be used by model-free approaches such as Model-Free Control [6] and Reinforcement Learning (RL) approaches [7].

RL led to groundbreaking advances in various domains of applications during the past decade, *e.g.* [8], [9]. However, the lack of formal stability guarantees for RL-based controllers [2] hinders their application on industrial safety-critical systems. Coupling advances in Machine Learning (ML) with long-established theories from the control community such as Lyapunov stability [10] could lead to a profitable trade-off. A terminology bridging the gap between control and RL communities is given in [11]. Recent studies led to promising perspectives, taking into account Lyapunov conditions directly in the learning process. A specific neural network approximating a Lyapunov function can be learned alongside an optimized control law [12], [13]. Unlike these closely related works, this study accounts for model uncertainties for each trajectory sample used in the learning process, updating the Lyapunov neural network within the optimization loop. Robustness can then be ensured as a stability certificate is obtained for the learned controller in presence of model uncertainties. Almost Lyapunov functions [14] are of particular interest in the context of numerical algorithms such as RL, as they rely on relaxed conditions compared to classical Lyapunov functions which are easier to verify numerically. Leveraging its ease of implementation and performance, Proximal Policy Optimization (PPO) [15] is considered for updating and optimizing the parameters of the neural networks. This method samples data by interacting with an unknown environment and using a gradient-based optimization of a surrogate objective function on small batches.

In this paper, a controller for attitude stabilization of a launcher upper stage module subject to sloshing is designed. The RL algorithm considered, based on PPO, automatically learns the control law with a trial and error process, adding Lyapunov stability conditions to ensure stability. The main contributions of this paper can be summarized as:

- Proposing a control design based on Deep Reinforcement Learning (DRL) to address the attitude control problem of a launcher upper stage module, in presence of uncertain nonlinear dynamics due to sloshing effect, saturation, time delay and bias in the actuation system. To the authors knowledge, no similar control solution with a stability certificate for an industrial space application has been developed.
- Proposing an extension of the PPO algorithm allowing to get a robust stability certificate, in presence of uncertainties.

The present work deals with a study case defined by the space industry. Hence, numerical values are subject to industrial confidentiality and can not be given in this paper regarding the parameters of the model and the simulations results (values on graphs).

The paper is organized as follows. Section 2 introduces the case of application, and launcher upper stage module's dynamics. Section 3 is devoted to the description of the

[1]Université Paris-Saclay, ONERA, Traitement de l'Information et Systèmes, 91123, Palaiseau, France. name.surname@onera.fr

[2]ArianeGroup SAS, 78130 Les Mureaux, France name.surname@ariane.group

[3]Now MaiaSpace, 92400 Courbevoie, France name.surname@maia-spaice.com

control algorithm based on elements from Lyapunov theory and Reinforcement Learning. Section 4 provides simulation results to illustrate the effectiveness of the proposed approach in terms of closed loop stability and in terms of robustness with respect to uncertainties.

II. Launcher Upper Stage Dynamics

During the exo-atmospheric phase of a launcher upper stage, short re-boosts can be performed with a re-ignitable main engine. Those will aim to correct the trajectory by reaching the desired equilibrium state, close-to-zero attitude angles and transverse acceleration while ensuring stability and mitigating perturbations such as the sloshing phenomenon which can appear in the tanks. The control input is the deflection angle β between the nozzle and the main axis of the launcher, which orientates the thrust. The thrust magnitude P is assumed constant. The control objective is to stabilize the attitude angle θ and the attitude rate $\dot{\theta}$ to zero. Figure 1 illustrates the parameters adopted in the planar representation of the problem. Let G be the center of mass (CoM) of the launcher, at a distance L_t from the thrust application point T. The CoM position is assumed constant during a re-boost. However, uncertainties considered in L_t can account for slight position changes. The transverse velocity of the launcher along the z_E axis of the body frame $(G; x_E, z_E)$ attached to the launcher is denoted $\dot{z}$. Two sloshing modes are considered to tackle potential interactions between modes, as in [6]. The equivalent mechanical model used for each mode is a pendulum [16], [17]. In Figure 1, A_i is the point of attachment of the pendulum i ($i = 1, 2$), B_i is the position of its corresponding mass m_i, $l_{p,i}$ the length, α_i the angle between x_E and pendulum i, and L_i the distance between A_i and G.

The natural damping factor of the sloshing mode i is defined by $\xi_i = \xi_i^{1g}\sqrt{\frac{g}{\gamma}}$, where g is the gravity constant, ξ_i^{1g} is the damping of the pendulum i under a 1 g gravity and $\gamma = \frac{P}{M+m_1+m_2}$ is the launcher's acceleration. Notation M refers to the mass of the launcher and I to its transverse inertia, excluding sloshing masses.

The dynamical model can be derived from classical laws of physics applied on the system presented in Figure 1. Linearized equations with a single pendulum can be found in [16] and [18]. We chose to derive here the full nonlinear equations without approximations. These equations are the following:

$$\begin{aligned} &l_{p_1}\ddot{\alpha_1} + \gamma\ \sin\alpha_1 + \ddot{z}\ \cos\alpha_1 + \ddot{\theta}\left(l_{p_1} - L_1\ \cos\alpha_1\right) \\ &- \dot{\theta}^2 L_1\ \sin\alpha_1 + 2\xi_1\sqrt{\gamma l_{p_1}}\dot{\alpha_1} = 0 \end{aligned} \tag{1}$$

$$\begin{aligned} &l_{p_2}\ddot{\alpha_2} + \gamma\ \sin\alpha_2 + \ddot{z}\ \cos\alpha_2 + \ddot{\theta}\left(l_{p_2} - L_2\ \cos\alpha_2\right) \\ &- \dot{\theta}^2 L_2\ \sin\alpha_2 + 2\xi_2\sqrt{\gamma l_{p_2}}\dot{\alpha_2} = 0 \end{aligned} \tag{2}$$

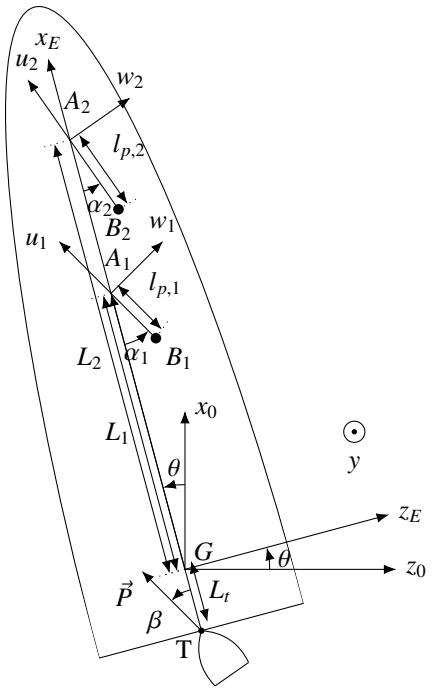


Fig. 1: Sloshing model with oscillating masses

$$\begin{aligned} &(M + m_1 + m_2)\,\ddot{z} + m_1 l_{p_1}\ddot{\alpha_1}\ \cos\alpha_1 + m_2 l_{p_2}\ddot{\alpha_2}\ \cos\alpha_2 \\ &+ \ddot{\theta}\left(m_1(l_{p_1}\ \cos\alpha_1 - L_1) + m_2(l_{p_2}\ \cos\alpha_2 - L_2)\right) \\ &- \dot{\alpha}^2\left(m_1 l_{p_1}\ \sin\alpha_1 + m_2 l_{p_2}\ \sin\alpha_2\right) + P\sin\beta = 0 \end{aligned} \tag{3}$$

$$\begin{aligned} &\left(I + m_1 L_1(L_1 - l_{p_1}\ \cos\alpha_1) + m_2 L_2(L_2 - l_{p_2}\ \cos\alpha_2)\right)\ddot{\theta} = \\ &m_1 L_1 l_{p_1}\ \cos\alpha_1\ddot{\alpha}_1 + m_2 L_2 l_{p_2}\ \cos\alpha_2\ddot{\alpha}_2 + (m_1 L_1 + m_2 L_2)\ddot{z} \\ &- \dot{\theta}^2\left(m_1 L_1 l_{p_1}\ \sin\alpha_1 + m_2 L_2 l_{p_2}\ \sin\alpha_2\right) + 2m_1\xi_1 l_{p_1}\sqrt{\gamma l_{p_1}}\dot{\alpha}_1 \\ &+ 2m_2\xi_2 l_{p_2}\sqrt{\gamma l_{p_2}}\dot{\alpha}_2 - PL_t\ \sin\beta \end{aligned} \tag{4}$$

The control and actuation system used in the upper stage module considered in this study case from industry induces some saturation, bias and time delay in the realization of the deflection angle β. By denoting β^c the commanded deflection angle, the following relation will be considered to model the effects of the control and actuation system:

$$\beta = \text{sat}\left(\beta^c(t-\tau)\right) + b_\beta \tag{5}$$

where sat(.) is the saturation function, τ the time delay and b_β the bias.

From equations (1)-(5), the system dynamics can be expressed as:

$$\dot{x}(t) = f\left(x(t), u(t), p\right) \tag{6}$$

With the state vector:

$$x = \begin{pmatrix}\theta & \dot{\theta} & \alpha_1 & \dot{\alpha}_1 & \alpha_2 & \dot{\alpha}_2 & \dot{z}\end{pmatrix}^T \tag{7}$$

and the control input $u = \beta^c$. The sloshing angles and rates are considered accessible. Although this is not trivial in practice, an observer can be designed as developed in [19], [20] with a Kalman Filter or a Sliding Mode Observer. The reconstruction of the states is not in the scope of this study.

The vector p gathers all the parameters used in the model that may suffer from uncertainty. It is defined as:

$$p = \begin{pmatrix} m_1 & m_2 & M & I & l_{p,1} & l_{p,2} & L_1 & L_2 & L_t & P \end{pmatrix}^T \quad (8)$$

The objective can now be stated as designing a control law mitigating sloshing while stabilizing the states to a desired value (set to zero here), in presence of uncertainties on the model parameters p.

The next section introduces some elements of control theory that will be used in the proposed RL algorithm to design a control law with stability certificate, in presence of uncertainties.

III. Lyapunov Control Theory and RL Algorithm

A Reinforcement Learning scheme will allow to learn an adapted controller for the problem. By following a policy $\pi : \mathcal{X} \rightarrow \mathcal{U}$, control actions will be taken according to the current state value. The objective of the policy is to minimize the cumulative discounted rewards for every state. Additionally, self-learned Lyapunov functions will be determined, along with the control policy, in order to obtain stability certificates. After a quick reminder on Lyapunov theory and a formulation of the conditions ensuring stability, the core of the Reinforcement Learning algorithm will be presented.

A. Control Theory preliminaries

Lyapunov theory is used here to verify stability [21]. We remind here the definition and properties of a stable system.
Definition 1: The origin is an equilibrium point stable for the system (6) if:

$$\forall \varepsilon \in \mathbb{R}^+, \exists\ \delta \in \mathbb{R}^+ s.t. \|x(0)\| \leq \delta \implies \forall t \geq 0, \|x(t)\| \leq \varepsilon$$

The system is asymptotically stable at the origin if it is stable at the origin and if $\exists \delta \in \mathbb{R}^+$ s.t. $\|x(0)\| \leq \delta \implies \lim_{t\rightarrow\infty} x(t) = 0$.

The functions considered are assumed to be Lipschitz continuous which helps training robust neural networks [22], i.e. networks giving same predictions if the input is perturbed imperceptibly.

Lie derivatives are introduced to determine the rate of the Lyapunov function along the given dynamics. For continuous time systems, they are defined as follows:
Definition 2: Consider the dynamical system (6) and a continuously differentiable function $v : \mathcal{D} \rightarrow \mathbb{R}$, with $\mathcal{D} \subseteq \mathbb{R}^n$. The Lie derivative of v over f is defined as:

$$L_f v(x) = \sum_{i=1}^{n} \frac{\partial v}{\partial x_i} \frac{dx_i}{dt} \quad (9)$$

Given these definitions, stability conditions derived from Lyapunov theory can be written. Computed for a fixed policy, Control Lyapunov functions (CLF) $V : \mathcal{X} \rightarrow \mathbb{R}^+$, with $\mathcal{X} \subseteq \mathbb{R}^n$ are continuously differentiable and should verify:

$$V(0) = 0 \quad (10)$$

$$\forall x \in \mathcal{X}\backslash\{0\}, V(x) > 0 \quad (11)$$

$$\forall x \in \mathcal{X}\backslash\{0\}, L_f V(x) < 0 \quad (12)$$

In the RL algorithm used in this paper, a neural network representing a Lyapunov function will be learned and act as a critic in an actor-critic framework. Due to the Reinforcement Learning scheme and the numerical aspect of the solution developed, Lyapunov conditions can only be evaluated at sampled states. Therefore, relaxed Lyapunov criteria are considered, as proposed with Almost Lyapunov functions introduced by [14].

Those can guarantee convergence even if the Lyapunov function may be not decreasing over some restricted subsets of the state space. Compared to classical Lyapunov theory, conditions (10)-(11) are kept unchanged but condition (12) is replaced by:

$$\forall x \in \mathcal{X}\backslash\Omega, L_f V(x) < 0 \quad (13)$$

with $\Omega = \{x \in \mathcal{X} | \dot{V}(x) \geq 0\}$ a subset of $\mathcal{X}$ "small enough", see [14].

In the RL framework considered here, the Lie derivative $L_f V$ will be approximated from the available sample values. Note that the RL algorithm does not require explicit knowledge of the system dynamics (model-free approach), but only numerical evaluation of a reward, and next observed state, corresponding to a given action (control input). Nevertheless, the system dynamics are used here to produce data for the reward evaluation (which corresponds to a simulated "environment", as named in RL algorithms). Given two consecutive states $x(t)$ and $x(t + \Delta t)$ separated by Δt, the numerical approximation of the Lie derivative is defined as:

$$L_{f,\Delta t} V(x) = \frac{1}{\Delta t} \left(V\left(x(t + \Delta t)\right) - V\left(x(t)\right) \right) \quad (14)$$

such that $\lim_{\Delta t\rightarrow 0} L_{f,\Delta t} V(x) = L_f V(x)$.

Closed loop stability of the system can hence be established under a given control policy if an Almost Lyapunov function can be found that satisfies the following decrease condition, numerically evaluated along the trajectories of the closed loop system. Hence, the condition (13) is replaced by:

$$\forall x \in \mathcal{X}\backslash\Omega, L_{f,\Delta t} V(x) < 0 \quad (15)$$

In the next section, the Deep Reinforcement Learning algorithm making use of these Lyapunov conditions to get stability certificates is presented.

B. Robust Deep Reinforcement Learning with Almost Lyapunov Functions

From here, in order to match algorithms from the literature, notations classically used in the Reinforcement Learning community will be used instead of the ones from control theory: the state x_t will be denoted by s_t and the control input u_t by the action a_t.

Based on the Proximal Policy Optimization algorithm [15] and works on safe deep learning from [12] [23] [24], the proposed algorithm makes use of Lyapunov stability conditions in the learning process. In this article, robustness with respect to uncertainties is furthermore requested. Lyapunov conditions are introduced as a Lyapunov risk in the loss function and the decrease condition is also taken into account for the advantage estimate computation. Compared to existing work of [12], the Lyapunov risk is evaluated at each optimization step, where parameters of the neural networks representing the policy, Lyapunov and value functions are updated. This difference with respect to [12] is introduced to handle uncertainties. Indeed, for this optimization, batches of trajectories obtained with different values of the parameter vector p are considered. Those are randomly generated in a set $\mathcal{P}$ corresponding to the convex hull of possible model parameters within the given ranges of uncertainties.

The main steps of the proposed approach are described in Algorithm 1. They are detailed below.

The policy π_μ, the value function V_μ^r and the Lyapunov function V_μ are represented by neural networks considering the update parameter μ. These networks are randomly initialized in the first step of the algorithm (line 1). Then, for each epoch, model parameters p are randomly chosen in $\mathcal{P}$ and used to generate system trajectories (of different dynamics, since using different model parameters, and under the current policy π_μ). A buffer B is then filled with those trajectories consisting in values of current state, action, reward and next state on a given temporal horizon (lines 2-3). The reward function r_t represents the improvement made when taking action a_t when the system is in state s_t and evolving to state s_{t+1}. It is defined here as:

$$r_t = 0.7 * (\theta^2 + \dot{\theta}^2) + 1.5 * (\alpha_1^2 + \alpha_2^2) + 0.2 * \dot{z}^2 \tag{16}$$

Then, optimization steps are performed to update the three neural networks by minimizing a loss function J with backpropagation (lines 4-11) over sample batches of size N taken from B (line 5). The different terms involved in the loss function are evaluated at each optimization step.

Based on the reward r_t, the advantage estimates $\hat{A}$ represent the estimate of the difference between the quality of a state-action combination (known as the Q-function) and the expected value in a given state (line 6). With a discount factor $0 < \gamma < 1$, advantage estimates are computed as:

$$\hat{A}(s_t, a_t) = \delta_t + \gamma\delta_{t+1} + \ldots + \gamma^{N-t+1}\delta_{N-1} \tag{17}$$

with:

$$\delta_t = r_t + \gamma V_\mu^r(s_{t+1}) - V_\mu^r(s_t) \tag{18}$$

To foster the validation of the Lyapunov decrease condition, the advantage estimates are modified, as proposed in [12], and are then denoted $\hat{A}_d^L$ with d (0< d <1) a fixed constant (line 7):

$$\hat{A}_d^L(s_t, a_t) = (1-d)\hat{A}(s_t, a_t) + d\ \min\left(0, -L_{f_{\pi_\mu},\Delta t}V_\mu(s_t)\right) \tag{19}$$

where $L_{f_{\pi_\mu},\Delta t}$ denotes the Lie derivative evaluated as in (14) along the closed loop trajectories obtained under the policy π_μ.

Algorithm 1 Robust Lyapunov Certified PPO (RLC-PPO)

1: Initialize policy network π_μ, the value function network V_μ^r, the Lyapunov function network V_μ and the buffer B (empty set)
2: **for** epochs k = 0, 1,...,K **do**
3: Add collected set of trajectories in B by running policy π_μ in the environment with random model parameters chosen in $\mathcal{P}$
4: **for** each optimization step **do**
5: Sample batches of size N from B
6: Compute advantage estimates $\hat{A}$:
$\delta_t \leftarrow r_t + \gamma V_\mu^r(s_{t+1}) - V_\mu^r(s_t)$
$\hat{A}(s_t, a_t) \leftarrow \delta_t + \gamma\delta_{t+1} + \ldots + \gamma^{N-t+1}\delta_{N-1}$
7: $\hat{A}_d^L \leftarrow (1-d)\hat{A}(s_t, a_t) + d\min\left(0, -L_{f_{\pi_\mu},\Delta t}V_\mu(s_t)\right)$
8: $J(\mu, d) \leftarrow \hat{\mathbb{E}}\Big[J^{CLIP}(\mu, d) - c_1 J^{VF}(\mu) + c_2 S_{\pi_\mu}(s_t) - c_{risk}R_{f,N,\Delta t}(\mu)\Big]$
9: Update the networks via AdamW: maximization of the loss with backpropagation on J
10: **end for**
11: **end for**

The general learning goal is defined as the loss function J. This objective function is expressed as (line 8):

$$J(\mu, d) = \hat{\mathbb{E}}\Big[J^{CLIP}(\mu, d) - c_1 J^{VF}(\mu) + c_2 S_{\pi_\mu}(s_t) - c_{risk}R_{f,N,\Delta t}(\mu)\Big] \tag{20}$$

Notation $\hat{\mathbb{E}}$ corresponds to the empirical average over the batches of samples. The constants c_1, c_2 and c_{risk} are positive coefficients chosen by the user that can be used to normalize the terms between them. Each term of equation (20) will be detailed further.

The next and final step of the algorithm (line 9) is the update of the neural networks with the stochastic gradient descent method AdamW and a backpropagation on the objective J. As noticed in [25], AdamW yields good performances with respect to the training loss and a better generalization than models trained with Adam. To take the robustness with respect to model parameters into account, the developed algorithm updates the policy, the value function and the Lyapunov function at the same time in the optimization loop. This leads to a more robust learning since Lyapunov functions need to change depending on the dynamics. As previously mentioned, accounting for robustness in the learning process of the control policy and the stability certificate is a contribution of this paper, compared to algorithms existing work such as [12] [13].

As proposed in [15], the objective function J considered in (20) takes into account the policy surrogate J^{CLIP} and a value function error term J^{VF}. The first term accounts for the policy surrogate, with a clipped objective function to avoid the new policy to be too different from the previous one. This can improve the possible sample efficiency issue due to the on-policy aspect, i.e. exploring by sampling actions according to the newest version of the policy, of the PPO

method. J^{CLIP} is defined as:

$$J^{\text{CLIP}}(\mu, d) = \hat{\mathbb{E}}_t \Big[\min \Big(c_r(\mu) \hat{A}_d^L, \text{clip}(c_r(\mu), 1-\varepsilon, 1+\varepsilon) \hat{A}_d^L \Big) \Big] \quad (21)$$

with $\hat{\mathbb{E}}_t$ the empirical average over one batch of samples, ε ($0< \varepsilon <1$) a hyperparameter [15] and the clipping ratio $c_r(\mu)$ avoiding to overshoot at each policy step defined by:

$$c_r(\mu) = \frac{\pi_\mu(a_t|s_t)}{\pi_{\mu_{old}}(a_t|s_t)}$$

The second term J^{VF} in (20) represents a mean-squared error on the value function $(V_\mu^r(s_t) - V_t^{target})^2$ with $V_t^{target} = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ the expected value obtained when evaluating the policy.

An entropy term S_{π_μ} to ensure sufficient exploration is also considered in the loss J as suggested in [26] [27].

A specific neural network determines the Lyapunov functions by guaranteeing the satisfaction of Lyapunov conditions. To do so, the learning steps aim to minimize a Lyapunov loss function defined as the Lyapunov risk [28] which will be the last term taken into account in the objective function J (line 8). Denoted $R_{f,N,\Delta t}(\mu)$, it can be expressed as:

$$R_{f,N,\Delta t}(\mu) = \frac{1}{N} \sum_{i=1}^{N} \Big(\max(0, -V_\mu(s_i)) + \max(0, L_{f,\Delta t} V_\mu(s_i)) \Big) + V_\mu^2(0) \quad (22)$$

This risk is positive and represents the satisfaction of Lyapunov conditions. When a Lyapunov function is found, the risk reaches its global minimum of zero. Its value is added in the loss function computation with a multiplicative positive constant c_{risk}.

IV. Simulation results

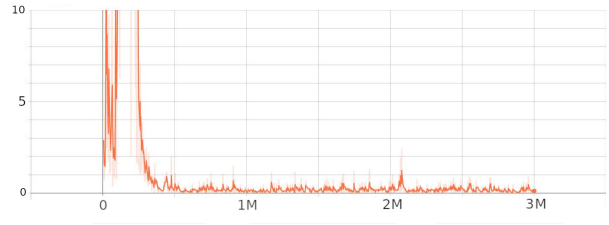
Simulation results are presented in this section to illustrate the stability property of the learned controller, in a robust way with respect to uncertainties regarding the model parameters. As stated in the introduction of the paper, numerical values used are subject to industrial confidentiality and thus cannot be given in this paper. The range of values used for initial conditions and uncertainties during the learning process and for the simulations will be given.

In the learning process, the simulated environment used to evaluate the reward and the next state makes use of the nonlinear dynamics (6) accounting for sloshing phenomenon and related parameter uncertainties, randomly chosen in $\mathcal{P}$, as well as time delay and bias on the realization of the commanded deflection angle β. A saturation is applied on the control.

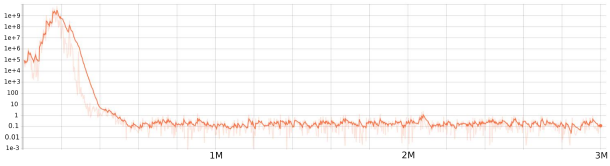
Tests led us to choose neural networks with 2 hidden layers of 4 neurons each, keeping their size relatively small. The actor, critic and Lyapunov networks use tanh as activation function on their hidden layers. A square activation function is used for the output layer of the Lyapunov network.

The Lyapunov risk, as evaluated during learning, is shown in Figure 2 in function of the number of steps. The value of 0.2 around which the risk converges is mainly due to the term related to the Lyapunov decreasing condition which is not verified on small subsets, as can also be noticed on Figure 4. Nevertheless, this risk convergence towards an almost zero value confirms that Lyapunov criteria are numerically verified (in the sense of almost Lyapunov functions) and thus that a stability guarantee is obtained.

The evolution of the loss function is also plotted in Figure 2 and converges towards a minimum close to zero confirming the algorithm found an optimal solution.



(a) Lyapunov risk



(b) Loss function

Fig. 2: Evolution of Lyapunov risk and loss function with respect to learning steps

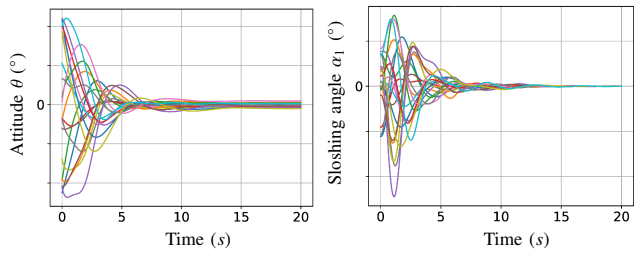
A. Closed loop stability

To illustrate the stabilizing property of the learned controller, closed loop trajectories are simulated, for a fixed given set of model parameter values, and from randomly generated initial conditions in θ, $\dot{\theta}$, α_1 and α_2. The order of magnitude of the range of initial conditions chosen for attitude and attitude rate are respectively $\pm 15°$ and $\pm 15°/s$.

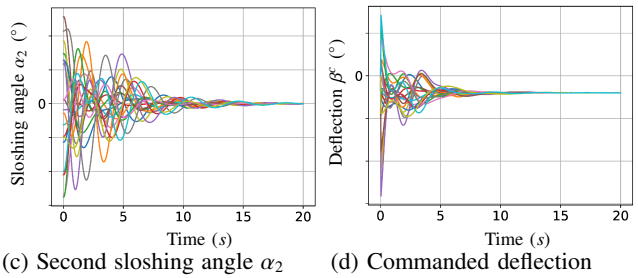
Sloshing initial angles have lower values, in a range of about $\pm 5°$. The re-boosts being considered after the long duration main boost, sloshing can be considered mitigated, justifying this small order of magnitude for the initial sloshing angles.

The results shown in Figure 3 demonstrate the stabilization of the system state from twenty different initial conditions. The primary goal of achieving near-zero sloshing angles is achieved, with both attitude and attitude rate converging to near-zero values. The commanded deflection angle converges to a value that compensates for the bias. Sloshing modes are mitigated, and the states stabilize to equilibrium in approximately 5 s, which meets industrial requirements for response time. The deflection angle remains within the requested range, avoiding saturation. Initial high-frequency variations of β are acceptable for this preliminary study.

Regarding stability certificate, the Lyapunov function learned by the algorithm in terms of a neural network can



(a) Attitude angle θ (b) First sloshing angle α_1



(c) Second sloshing angle α_2 (d) Commanded deflection angle β^c

Fig. 3: Closed loop trajectories obtained with the learned controller

be evaluated and represented in the $(\theta, \dot{\theta})$ subspace of the state. The same closed loop trajectories as in Figure 3 are also represented in this phase portrait for the attitude. These two representations are proposed in Figure 4.

The color bar on the right part of the figure represents the values of the Lyapunov function. As can be seen, closed loop trajectories converge towards a neighborhood of the origin. A small attitude static error is observed due to the use of robust learning (i.e. learning over different possible parameters values in $\mathcal{P}$ to cope with model uncertainties) which leads the algorithm to converge towards an almost zero cost.

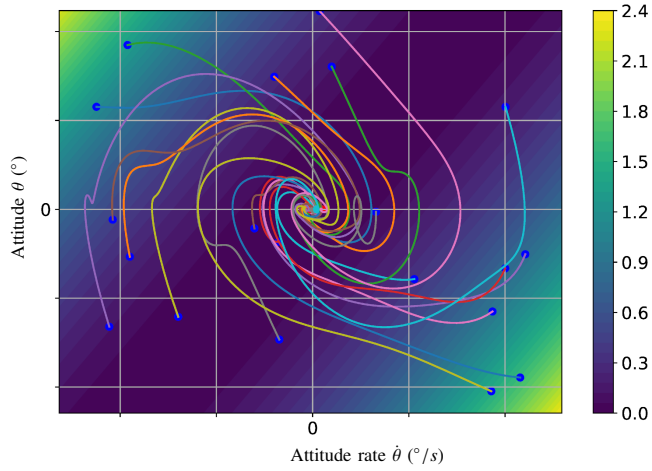


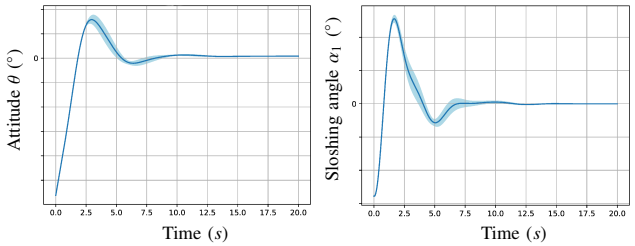
Fig. 4: Closed loop trajectories in attitude phase portrait and values of the learned Lyapunov function (color bar)

As can also be noticed, there exist some finite subsets inside which the value of the Lyapunov function is non-decreasing along the closed loop trajectories of the system. Nevertheless, by exploiting the framework of Almost Lyapunov functions developed in [14], stability guarantees can be possibly obtained from a theoretical point of view. To do so, subsets Ω used in relaxed condition (13), where the Lyapunov function is non-decreasing, should be properly characterized. This point is beyond the scope of this paper and will be addressed in future work.

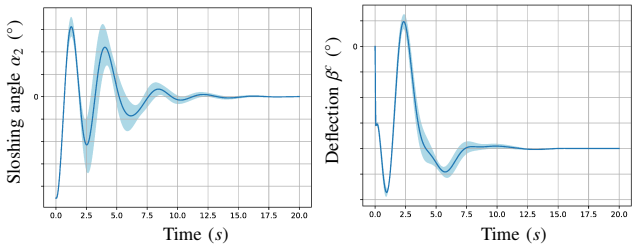
B. Robustness with respect to parameters uncertainties

To illustrate the robustness of the learned controller against model uncertainties, 200 closed-loop trajectories were simulated, each with different randomly generated dynamic model parameters within $\mathcal{P}$. Due to industrial confidentiality, only orders of magnitude are provided. The range of uncertainties is set to 10 % of the nominal value for the model parameters except for the pendulum characteristics which consider a 20 % uncertainty range. The same initial conditions are considered for these trajectories, to solely illustrate here robustness with respect to parameters uncertainties.

Figure 5 presents the mean (solid dark blue curve) and the standard deviation around the mean (light blue shaded envelope) over these 200 trajectories. The attitude phase portrait in Figure 6 shows the 200 simulated closed-loop trajectories. A single set of initial conditions is used for clarity, but similar plots can be provided for the range of initial conditions used during the learning process. As observed in these figures, the learned controller guarantees stability despite variations in the model parameters, illustrating the robustness of the proposed approach with respect to uncertainties.



(a) Attitude angle θ (b) First sloshing angle α_1



(c) Second sloshing angle α_2 (d) Commanded deflection angle β^c

Fig. 5: Mean and standard deviation over 200 closed loop trajectories with different model parameters values

V. Conclusion

In this paper, a Deep Reinforcement Learning algorithm has been proposed to design a control law for a launcher

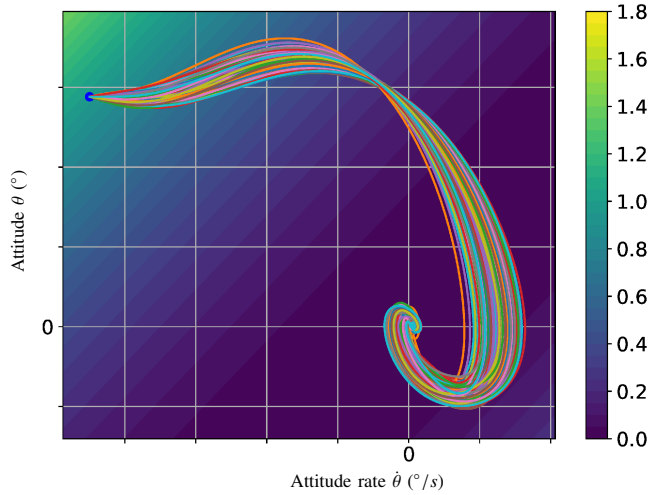


Fig. 6: Attitude phase portrait representation of 200 closed loop trajectories with different model parameter values, and values of the learned Lyapunov function (color bar)

upper stage module subject to the sloshing effect. Nonlinear dynamics of the system are accounted for as well as saturation, bias and time delay on the control input. Using conditions derived from Lyapunov theory, the actor-critic PPO algorithm has been modified in order to learn a robust controller, in presence of model uncertainties, along with a stability certificate. Simulation results have been provided to illustrate the good performance of the learned controller, in terms of time of response, stability and robustness.
However preliminary, these first results are hence encouraging to consider possible future applications, since this robustness of stability condition is of paramount importance to guarantee a safe evolution of a launcher upper stage module, in presence of real-world perturbations such as the ones due to the sloshing phenomenon.

Further studies will include constraint handling and comparison of the proposed approach with other nonlinear robust controllers from the control theory literature.

References

[1] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059.

[2] G. Waxenegger-Wilfing, K. Dresia, J. C. Deeken, and M. Oschwald, "A reinforcement learning approach for transient control of liquid rocket engines," *arXiv preprint arXiv:2006.11108*, 2020.

[3] A. Bourdelle, J.-M. Biannic, B. Laurent, H. Evain, C. Pittet, and S. Moreno, "Modeling and control of propellant slosh dynamics in observation spacecraft with actuators saturations," in *EUCASS 2019*, 2019.

[4] C. Jetzschmann, H. Strauch, and S. Bennani, "Model based active slosh damping experiment," *arXiv preprint arXiv:1801.10017*, 2018.

[5] P. Apkarian and D. Noll, "Nonsmooth H_∞ synthesis," *IEEE Transactions on Automatic Control*, vol. 51, no. 1, pp. 71–86, 2006.

[6] P. Cocaul, H. Piet-Lahanier, M. Ganet, S. Bertrand, and L. Lemazurier, "Anti sloshing control law design for launchers using a model-free approach," 9^{th} *European Conference For Aeronautics And Space Sciences (EUCASS)*, 2022.

[7] M. Ganet and P. Cocaul, "New perspectives for launcher gnc: Machine learning for upper stage sloshing control," *AAS American Astronomical Society*, 2022.

[8] B. Luo, H.-N. Wu, and T. Huang, "Off-policy reinforcement learning for h_∞ control design," *IEEE transactions on cybernetics*, vol. 45, no. 1, pp. 65–76, 2014.

[9] E. Bøhn, E. M. Coates, S. Moe, and T. A. Johansen, "Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 523–533.

[10] A. M. Lyapunov, "The general problem of the stability of motion," *International journal of control*, vol. 55, no. 3, pp. 531–534, 1992.

[11] B. Recht, "A tour of reinforcement learning: The view from continuous control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 253–279, 2019.

[12] Y.-C. Chang and S. Gao, "Stabilizing neural control using self-learned almost Lyapunov critics," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1803–1809.

[13] W. Jin, Z. Wang, Z. Yang, and S. Mou, "Neural certificates for safe control policies," *arXiv preprint arXiv:2006.08465*, 2020.

[14] S. Liu, D. Liberzon, and V. Zharnitsky, "Almost Lyapunov functions for nonlinear systems," *Automatica*, vol. 113, p. 108758, 2020.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[16] J. M. Adler, M. S. Lee, and J. D. Saugen, "Adaptive control of propellant slosh for launch vehicles," in *Sensors and sensor integration*, vol. 1480. International Society for Optics and Photonics, 1991, pp. 11–22.

[17] A. Shekhawat, C. Nichkawde, and N. Ananthkrishnan, "Modeling and stability analysis of coupled slosh-vehicle dynamics in planar atmospheric flight," in *44th AIAA Aerospace Sciences Meeting and Exhibit*, 2006, p. 427.

[18] A. G. de Souza, L. C. de Souza *et al.*, "Satellite attitude control system design taking into account the fuel slosh and flexible dynamics," *Mathematical Problems in Engineering*, vol. 2014, 2014.

[19] A. G. de Souza and L. C. de Souza, "Design of satellite attitude control system considering the interaction between fuel slosh and flexible dynamics during the system parameters estimation," *Applied Mechanics and Materials*, vol. 706, pp. 14–24, 2015.

[20] B. Bandyopadhyay, P. Gandhi, and S. Kurode, "Sliding mode observer based sliding mode controller for slosh-free motion through pid scheme," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 9, pp. 3432–3442, 2009.

[21] H. Khalil, "Nonlinear systems, printice-hall," *Upper Saddle River, NJ*, vol. 3, 1996.

[22] P. Pauli, A. Koch, J. Berberich, P. Kohler, and F. Allgöwer, "Training robust neural networks using lipschitz bounds," *IEEE Control Systems Letters*, vol. 6, pp. 121–126, 2021.

[23] M. Han, L. Zhang, J. Wang, and W. Pan, "Actor-critic reinforcement learning for control with stability guarantee," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6217–6224, 2020.

[24] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," *Advances in neural information processing systems*, vol. 30, 2017.

[25] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[26] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[27] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[28] Y.-C. Chang, N. Roohi, and S. Gao, "Neural Lyapunov control," *Advances in neural information processing systems*, vol. 32, 2019.